Plan Representation

EUROPA Compared with Classical Planning

EUROPA was designed to support planning for complex systems, such as spacecraft and rovers. This page introduces the EUROPA planning paradigm and explains how it differs from classical approaches.

At a very high level, planning can be considered a process of generating descriptions of how to operate some system to accomplish something. The resulting descriptions are called *plans*, and the desired accomplishments are called *goals*. In order to generate plans for a given system a *model* of how the system works must be given. For example, given a model of a spacecraft, and the goal of collecting and transmitting experimental data, the resulting plan would describe the operations needed to gather and send the data.

EUROPA, as is the case with its predecessors, uses declarative descriptions of models, plans and goals. This means that a single planner can be applied to different systems and problems, simply by providing different models and goals. For such declarative planners[1] the expressiveness of the languages for models, goals and plans is of great importance.

In traditional planning, models are described in terms of a set of Boolean state indicators (called *fluents*) and a set of actions that can modify the fluents. Goals are given in terms of an initial specification of fluents which are true or false and a set of different fluent values to achieve. A plan is then a sequence of actions that modifies the initial state to get the desired values.

For example, see Figure 1 which depicts a sequence of plan states (fluents that are false are colored grey). Each plan state is modified by actions to *effect* a new plan state. Actions can have *preconditions* e.g. one can only *GrabKey* if *HaveKey* is false. Actions have *effects* which are fluents arising in the succeeding plan state.

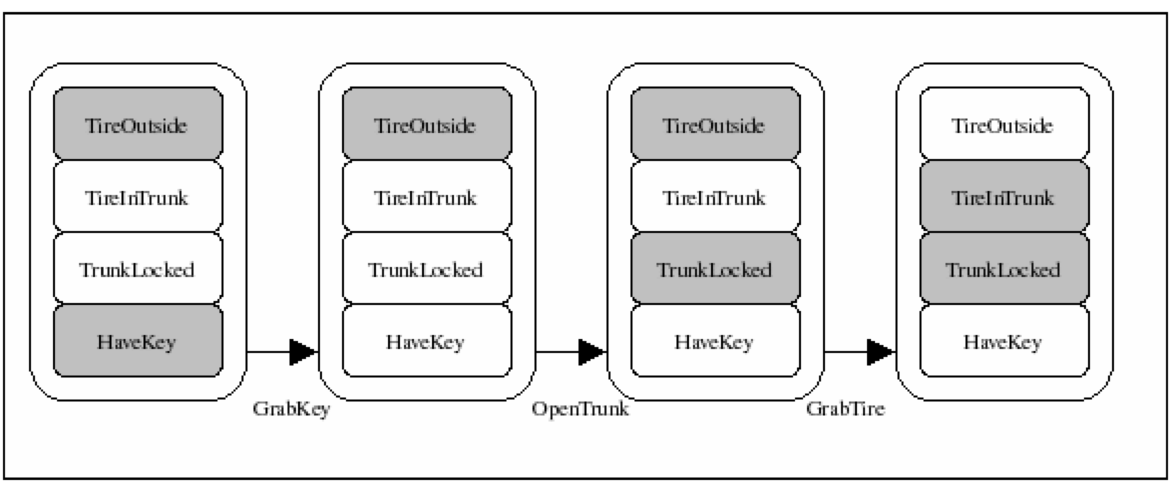

Figure 1: The Tire-World Domain - an example of a sequential operator plan. States contain fluents that are true (white) or false (greyed-out). Actions effect plan state.

This basic traditional approach is too simplistic for real-world applications. Consider a complex system like the Deep Space One spacecraft. It consists of multiple interacting subsystems, such as navigation, attitude control, camera, and main engine. Multiple subsystems may be active at the same time, each performing a different activity. There are also interactions among the subsystems, whether or not one or both are active. The camera, for example, cannot be taking a picture while the main engine is running. Finally, the classical assumptions concerning the relations between activities and states do not hold in many cases. Actions may have durations and may occur in specific time windows. States that are often considered passive, such as pointing at a target, are in fact activities. To plan for such complex systems, the expressiveness of model, plan and goal descriptions must be increased significantly.

Temporally Scoped Predicates (Tokens)

A first divergence from classical planning is an absence of explicit semantic differences between actions and states. EUROPA uses the same predicate logic formalism to describe states and actions and makes no distinction between the two. Predicates are used to describe things that are true. No facility is available to describe things that are false. Instead, we rely on scoping the temporal extent over which a predicate holds.[2]

Figure 2: Temporally scoped predicates for actions and states.

For example, Figure 2 depicts a fragment from the tire-world example. The fluent *HaveKey* being false is represented with the predicate *DoNotHaveKey*. The action *GrabKey* is also described with a predicate. The predicates are laid out in a sequence corresponding to the times over which they hold. Each predicate is annotated with an interval indicating the *temporal extent* over which it holds. Time is discretized into integers. Since the formalism is based on predicate logic, predicates can have arguments. For example, a *tireLocated* predicate might include an argument for the specific location of the tire i.e. *Locations = {Trunk, Ground}*. Thus one could state *tireLocated(Trunk)* or *tireLocated(Ground)*. In EUROPA, an instance of a temporally scoped predicate is called a *Token*. Three of the five built in variables on a token describe when it starts, ends, and what its duration is. These variables are unsurprisingly called *start*, *end* and *duration* respectively. The relationship *start + duration = end* is maintained as an invariant.

Token Relationships

In EUROPA, *Tokens* take the place of actions and fluents in classical planning. A second departure from classical planning is the increased range of relationships which can be expressed in EUROPA. Classical planning is restricted to talk about preconditions and effects[3]. Since Tokens contain variables for temporal scope and since EUROPA is based on an underlying constraint-programming model, relationships between tokens are naturally expressed as simple arithmetic constraints over these *temporal variables*. Following Allen?s relations (Allen, 1983), Figure 3 illustrates the range of relationships which can be expressed between any token pair. In each case, a qualitative description is provided and the underlying semantics are indicated as constraints on token time-points.

Figure 3: Temporal Relations between Tokens

---

1. Declarative planners are in contrast with special-purpose planners, in which the system model is hard-coded into the planner.
2. Intervals have long been used in Temporal Planning to represent actions and states with duration.
3. This has been expanded in some recent planning work.